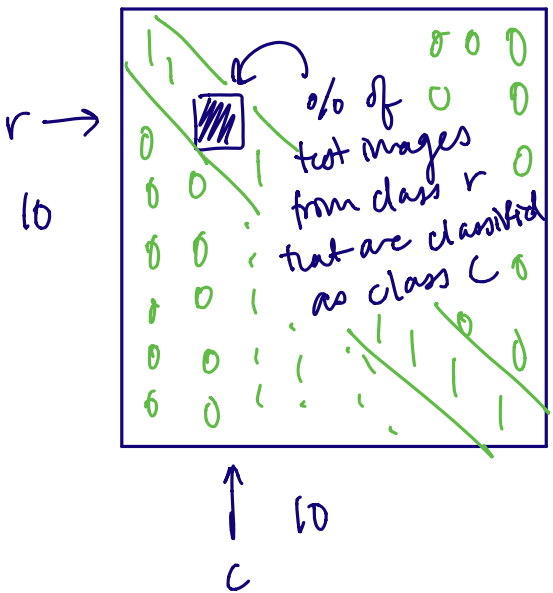
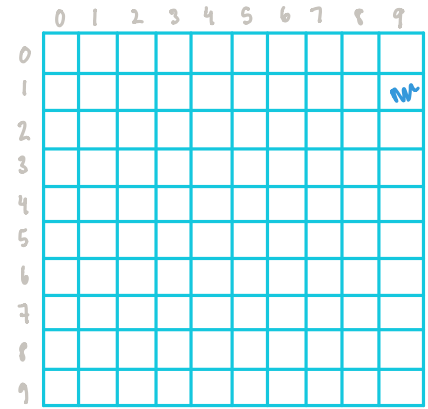
Confusion Matrix





how many test images from class 1 were defined as class 9?

Confusion Matrix [1][9] = # of class 1 images defined as 9 / # of class 1 images

> find indexes of "i" class images in testLabels (indexesOfLabels)

> go through each index, if value of results at that index == "j", increment counter (valueCounter)

> populate confusionMatrix [i][j] with (valueCounter / indexesOfLabels.size()) 100.

Naive Bayes Outline

Tasks

- Read training data from files and generate a model } Training stage
- Save a model as a file
- Load a model from a file } Classifying stage
- Classify images from a file

ⓒ ReadData class

- parse data file

ⓜ Model class

name of
- class variable
- prior vector
- probability vector 31

- find $P(F_{00} = 0 \mid class = "1")$, $P(F_{00} = 1 \mid class = "1")$
  $P(F_{01} = 0 \mid class = "1")$, $P(F_{01} = 1 \mid class = "1")$
  $P(F_{02} = 0 \mid class = "1")$, $P(F_{02} = 2 \mid class = "1")$
  ⋮
  $P(F_{2828} = 0 \mid class = "1")$, $P(F_{2828} = 1 \mid class = "1")$

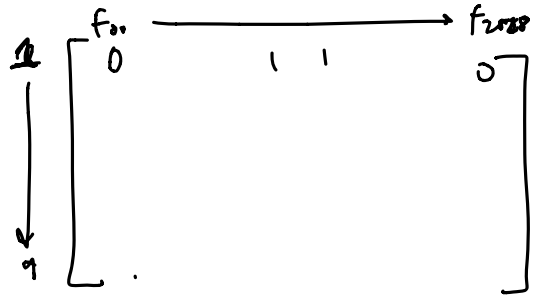
(generate array model of probabilities for class 1)

model possibilities

> 2D Vector (2 x 784) (9 x 784)

> 3D Vector (28 x 28 x 2) (28 x 28 x 9)

Model (1)



> priors : $P(class = c) = \frac{\#\text{ of training examples where } class = c}{\#\text{ of training examples}}$

$P(class = 1) = \frac{\#\text{ of training examples where class} = 1}{\#\text{ of training examples}}$

prior vector = [0.~, —, —, —, —, —, —, —, —]
0 1 2 3 4 5 6 7 8

all prior probabilities

Save model as model instance

⓪ Read Model Class ⟶ 2D vector

- class 1 probability =

$$\log(P(\text{class}_1)) + \log(P(f_{1,1}|\text{class}_1)) + \log(P(f_{1,2}|\text{class}_1)) + \ldots + \log(P(f_{28,28}|\text{class}_1))$$

prior — probabilities

- $P(f_{i,j} \mid \text{class } 1) =$ ~

posterior probabilities vector [~ , ----------]

0 ... 8

- highest values is the correct class!

Steps : training

> start with "ZERO"

> using "training labels", find indexes of all ZERO images ← store in: indexes_of_images []

> using "training images", start with first pixel in each image @ indexes

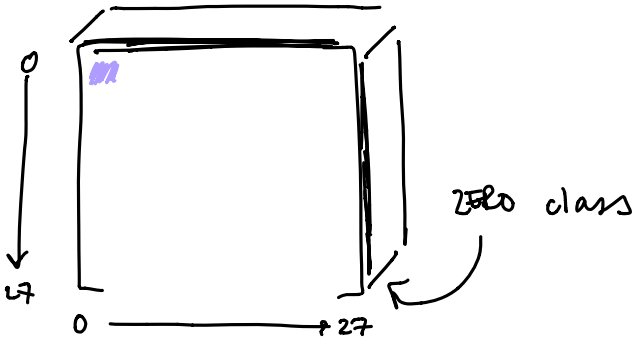
- if pixel = '#' or '+', increment on Counter.
- length of indexes_of_images is total # of examples where class = ONE.

~~~~ = $P(F_{i,j} = f \mid class = c) = \dfrac{k + \#\text{ of times } F_{i,j} = f \text{ when } class = c}{2k + \text{Total number of training examples where } class = c}$

~~~~ : 0,0 — 1 — ZERO — arbitary k — onCounter — indexes_of_images.length

> populate model Array with probability found :

model Array [ZERO] [0] [0] = ~~~~



> repeat process for all pixels

> repeat process for all digits

This will result in a model with all pixels/levels filled.

* write this model to a file to save it.

Steps : classifying

* what you have : file of images, textfile with model

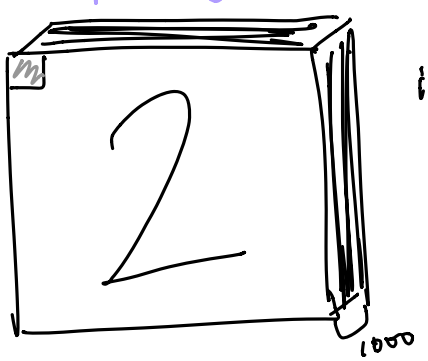
> read model.txt and populate probabilityVector

> read "testImages.txt" and populate testImagesVector

> posteriorProbabilityVector is float with 10 spots

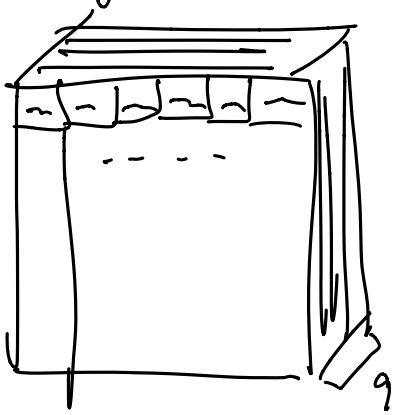
> PPV[digit] = priorVector[digit] + probability[digit][0][0] + ... + probability[digit][28][28]

posterior probability: $\log(P(class)) + \log(P(f_{1,1}|class)) + \log(P(f_{1,2}|class)) + \ldots + \log(P(f_{28,28}|class))$



if (m = # || +) → probabilityVector[digit][m]

else → 1 − probabilityVector[digit][m]



> start @ first pixel

> check if on/off

- if (m = # || +) → probabilityVector[digit][m]

  else → 1 − probabilityVector[digit][m]

© Detector Class
- testImagesVector
- probabilityVector
- posteriorProbabilityVector
- priorVector

> start with first image in testImages

Trainer Class

- training_images [][][]
- training_labels []
- priors []
- model [][][]
- numberOfTimesDigitOccurs []

> getters for all variables (not # of times)

initializeTrainer
- > setTrainingImages (file)
  - ↳ put stuff from file into training_images
- > set TrainingLabels (file)
  - ↳ put stuff from file into trainingLabels
- > set #oftimes digit occurs (trainingLabels [])
  - ↳ populate numberOfTimesDigitOccurs []
- > set priors (# of times Vector)
  - ↳ populate priors
- > set Model (trainingLabels, trainingImages)
  - ↳ populate model

Detector Class

- test_images [][][]
- test_labels []
- posteriors [][]
- results [] → size = 1000
- confusion_matrix []
- model [][][] } pass into constructor
- priors []

> getters for all variables

generateResults
- > set Test Images (file)
  - ↳ put stuff from file into test_images
- > set Test Labels (file)
  - ↳ put stuff from file into test_labels
- > set Posteriors (testImages)
  - ↳ populate posteriors for every page
- > set Results (posteriors)
  - ↳ calculate max of every row of posteriors, populate results array
- > set Confusion Matrix (testlabels, results)
  - ↳ compare and populate